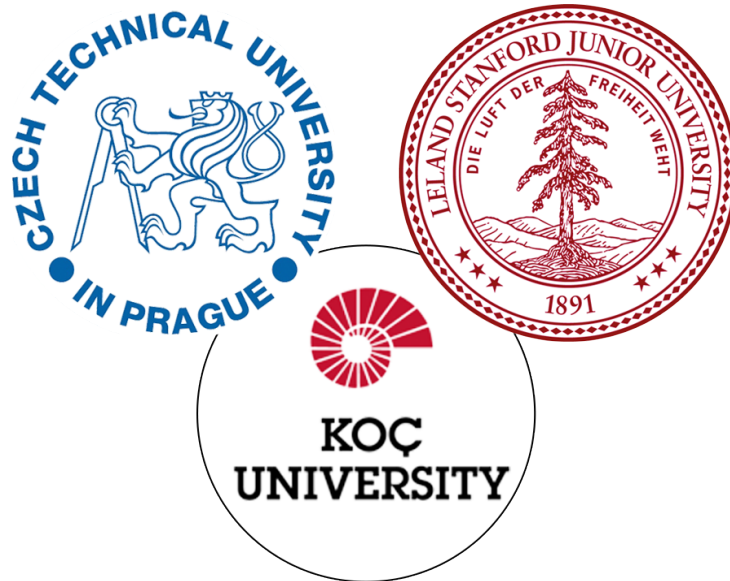


CS Bridge, Lecture 3

More Karel Control Flow

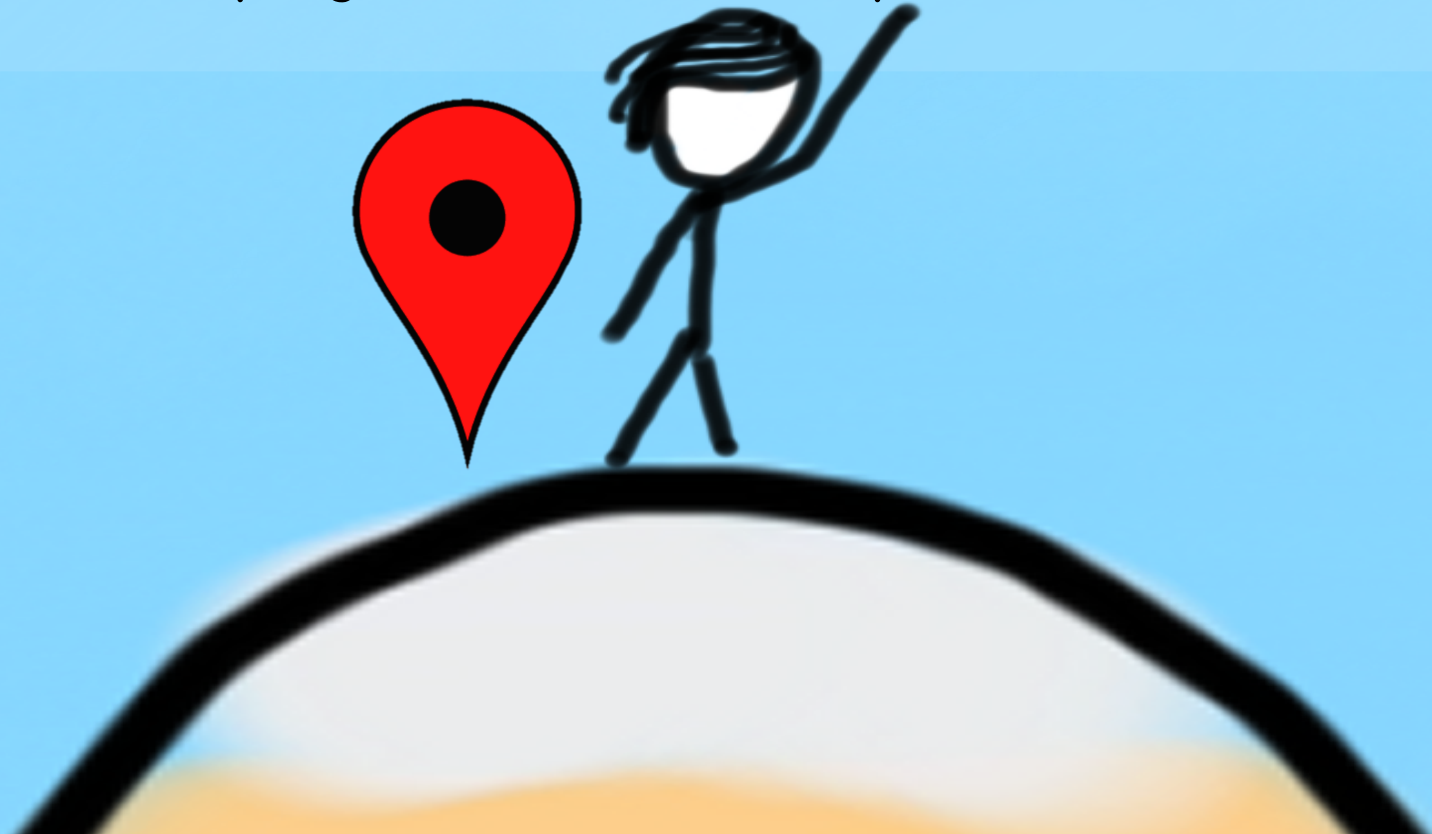


Lecture Plan

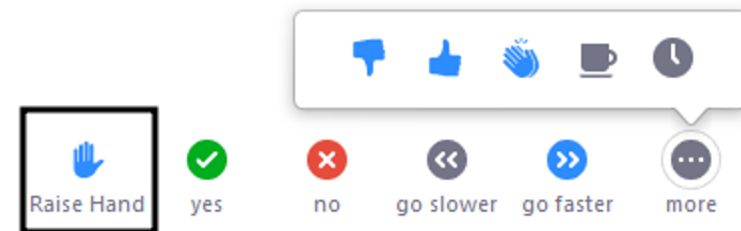
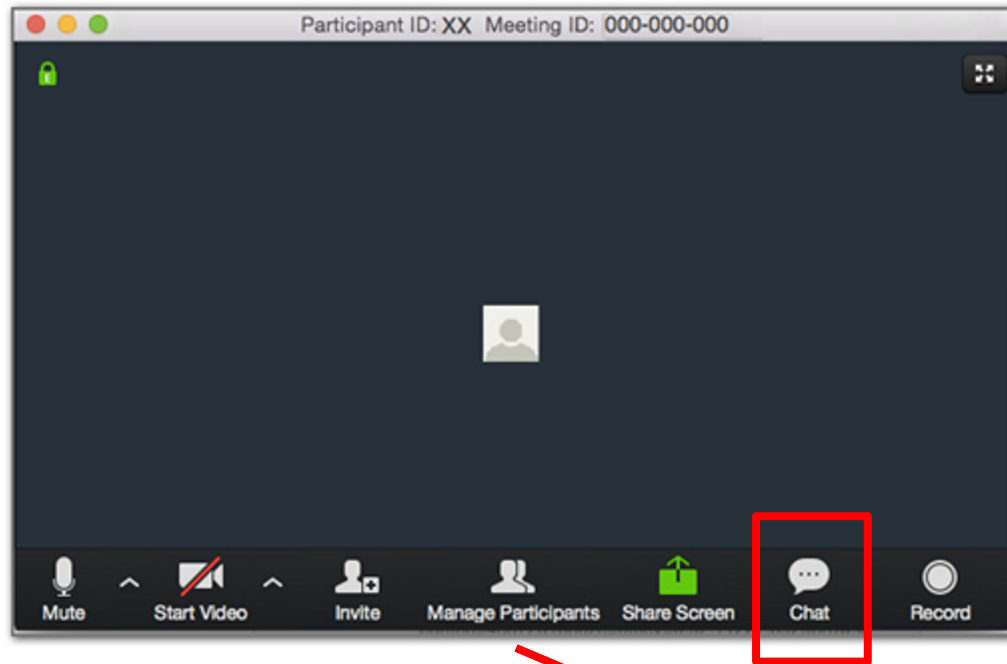
- **Review:** Karel and Control Flow
- If/Else Statements
- **Practice:** Hurdle Jumper
- Decomposition and Top-Down Design

Learning Goals

1. Code using conditions
2. Trace programs that use loops and conditions



Ask Questions!



Lecture Plan

- **Review:** Karel and Control Flow
- If/Else Statements
- **Practice:** Hurdle Jumper
- Decomposition and Top-Down Design

Control Flow

Control Flow lets us control the “flow” of our Karel program.

- Example: repeat something 5 times
- Example: repeat something until Karel is blocked

Control Flow: For Loops

```
for i in range(count):  
    statement  
    statement  
    ...
```

Repeats the statements in the body *count* times.

Control Flow: While Loops

```
while condition:  
    statement  
    statement  
    ...
```

Repeats the statements in the body until *condition* is no longer true. Each time, Karel executes *all statements*, and **then** checks the condition.

Possible Questions

`front_is_clear()`

`beepers_present()`

`beepers_in_bag()`

`left_is_clear()`

`right_is_clear()`

`facing_north()`

`facing_south()`

`facing_east()`

`facing_west()`

`front_is_blocked()`

`no_beepers_present()`

`no_beepers_in_bag()`

`left_is_blocked()`

`right_is_blocked()`

`not_facing_north()`

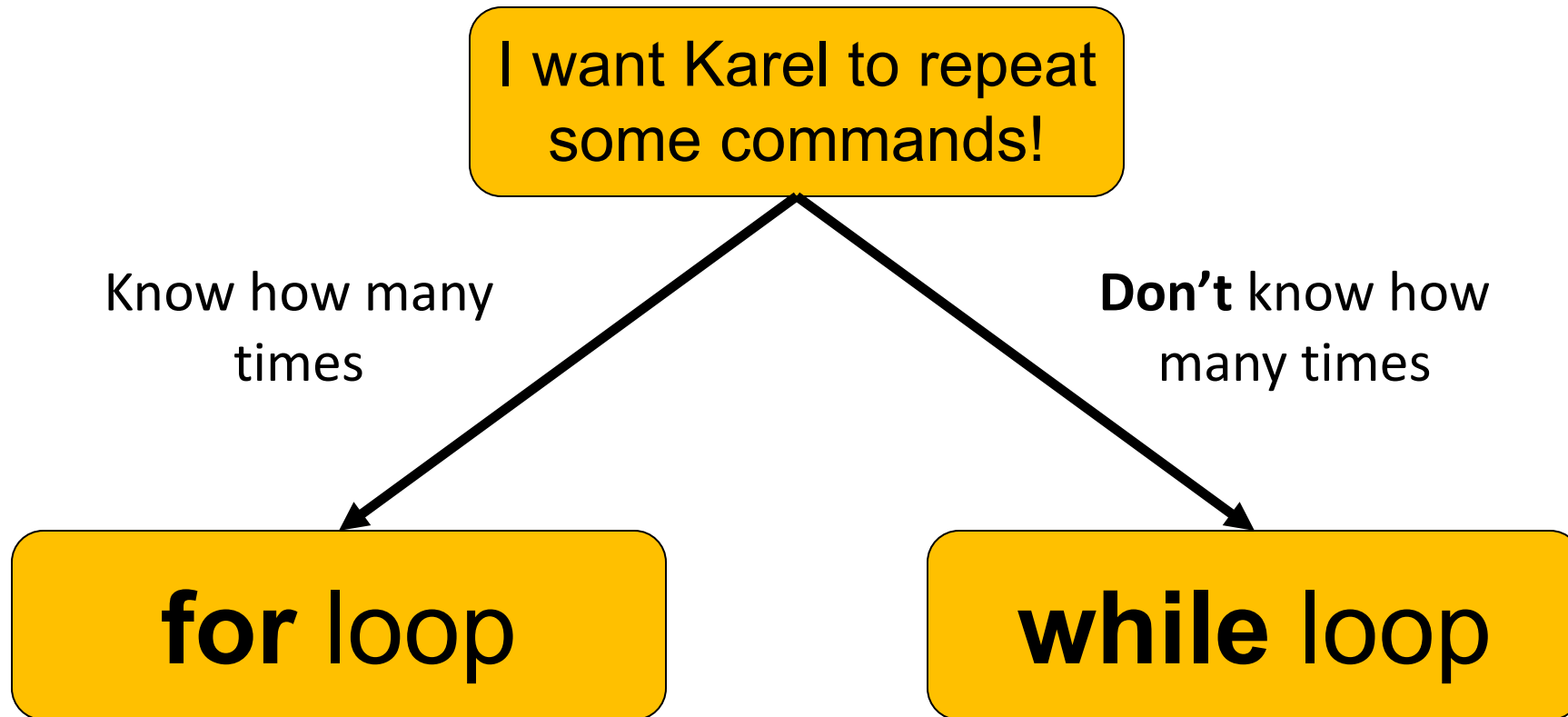
`not_facing_south()`

`not_facing_east()`

`not_facing_west()`

This is taken from the [Karel Reference](#).

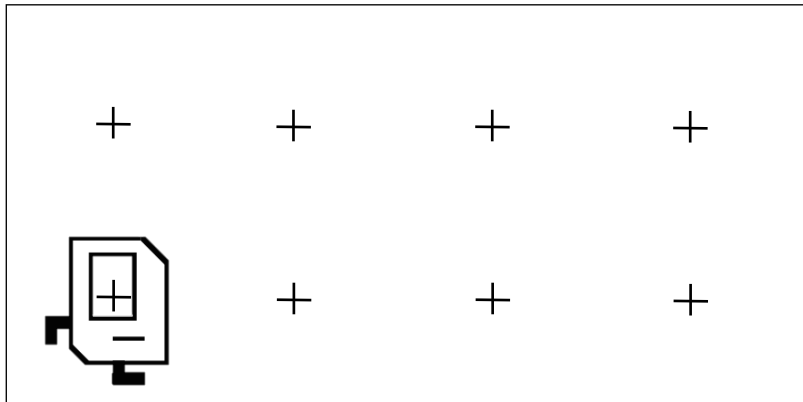
Loops Overview



Fencepost

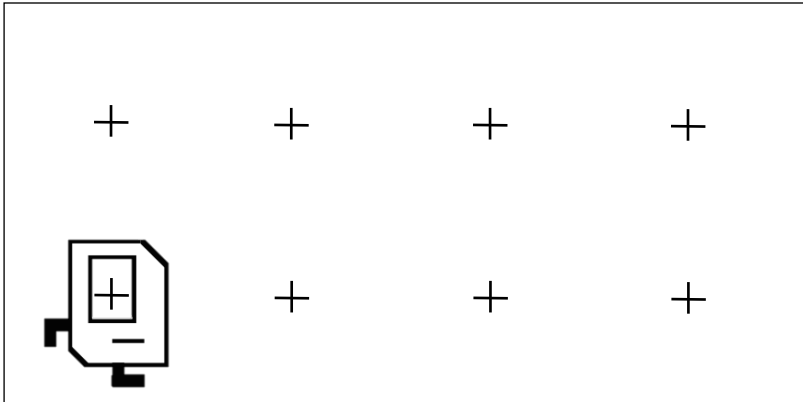
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?

Let's first pretend the program must only work for this world.
What commands should Karel execute?



Fencepost

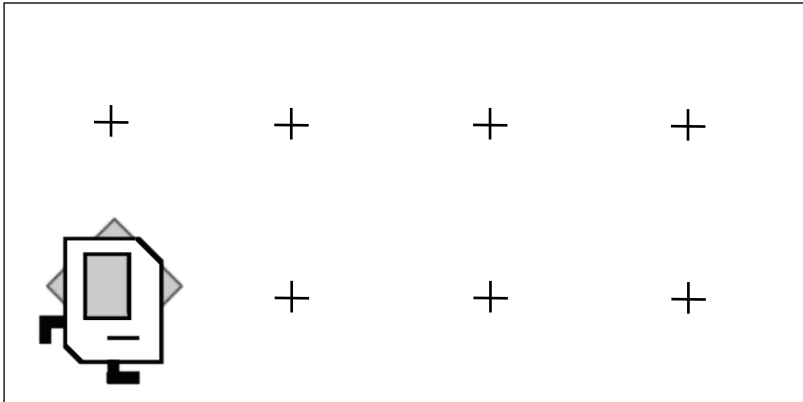
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

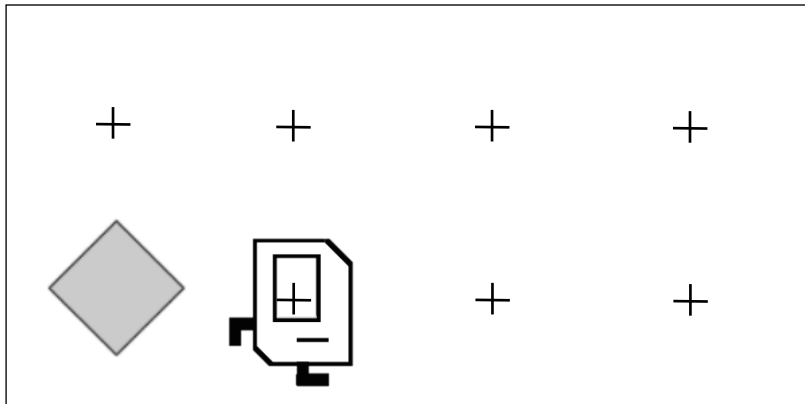
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()
```

```
move()
```

```
put_beeper()
```

```
move()
```

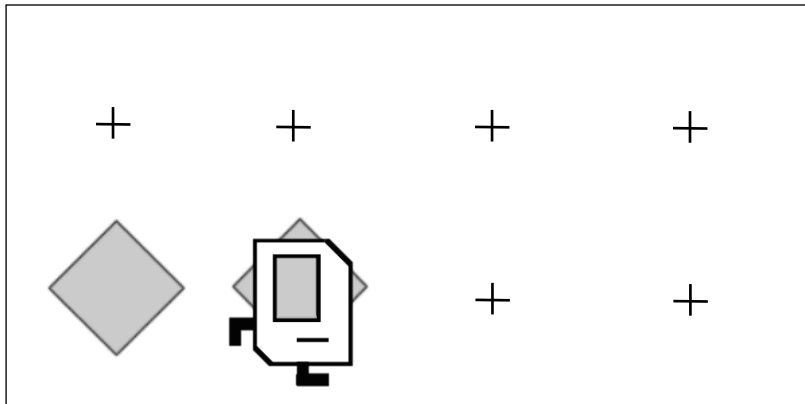
```
put_beeper()
```

```
move()
```

```
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()
```

```
move()
```

```
put_beeper()
```

```
move()
```

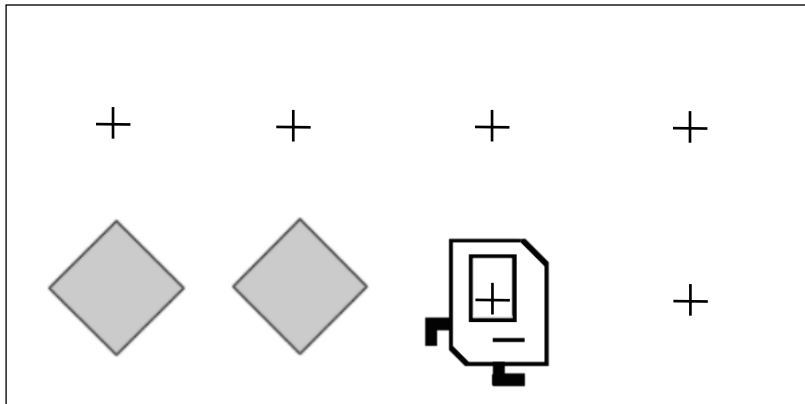
```
put_beeper()
```

```
move()
```

```
put_beeper()
```

Fencepost

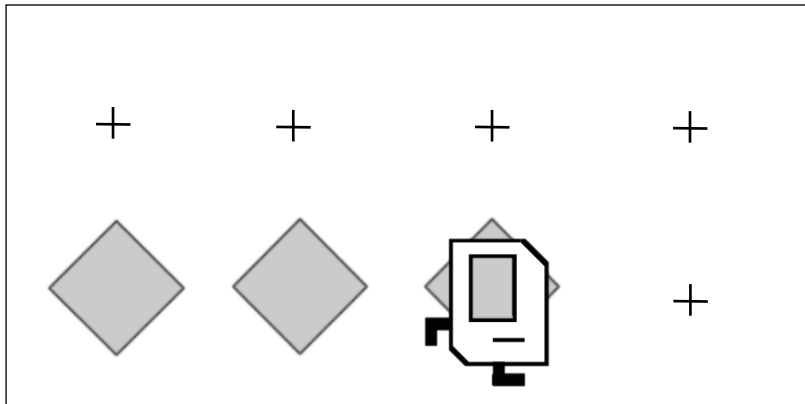
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```


Fencepost

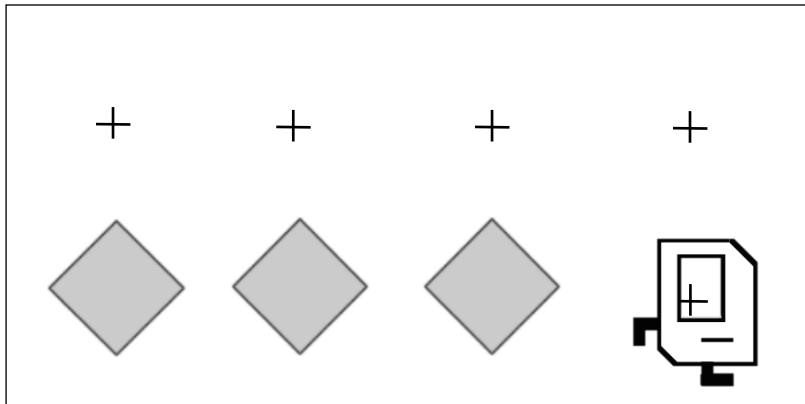
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

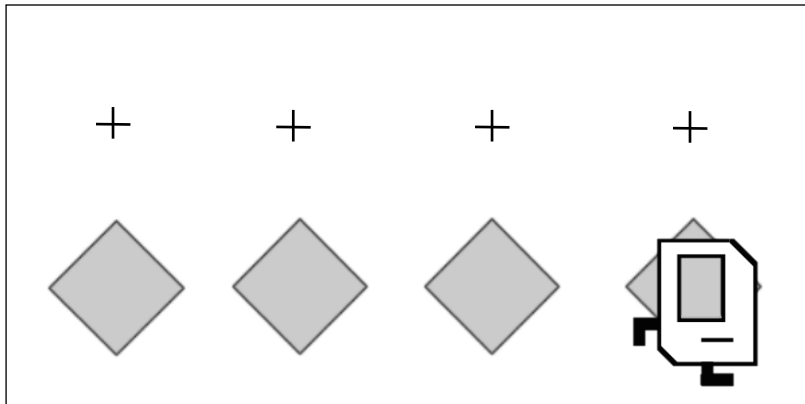
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

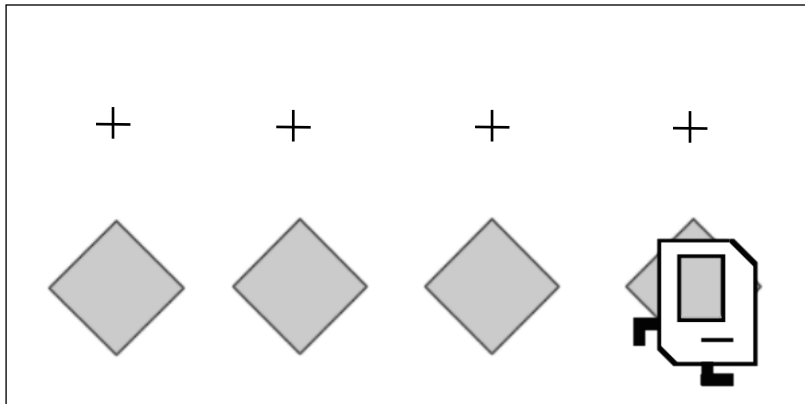
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put beeper()
```

Fencepost

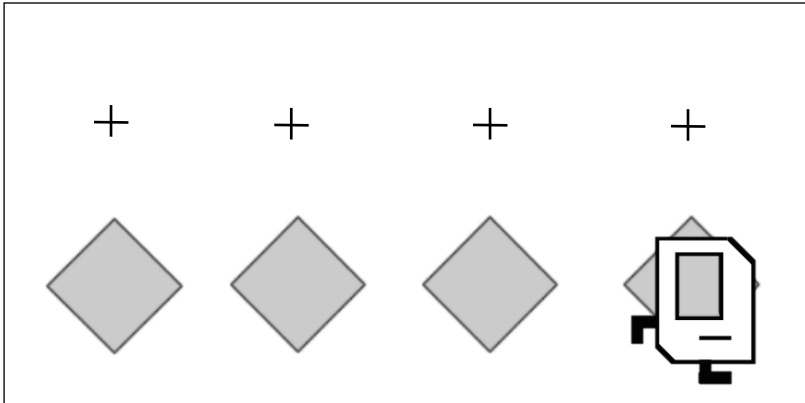
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

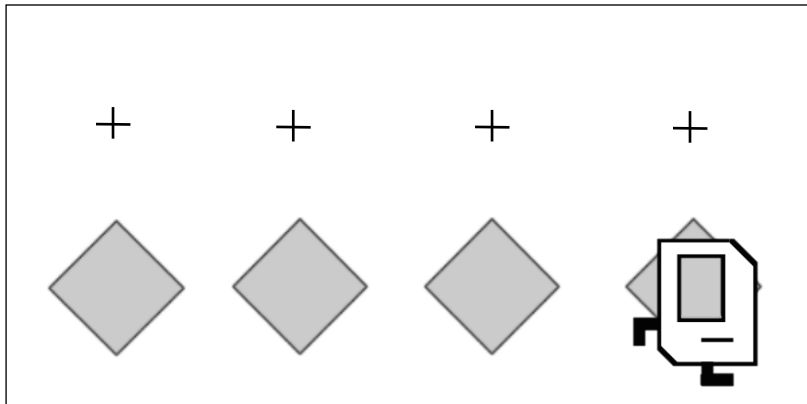
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?

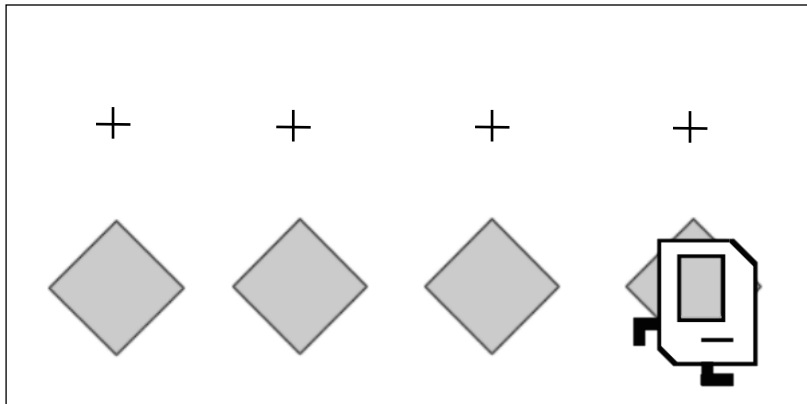


We must put 4
beepers but
move 3 times!

```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?

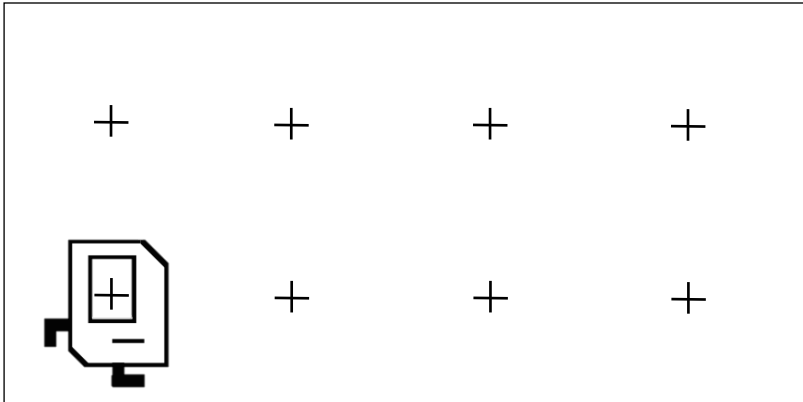


We must put 4
beepers but
move 3 times!

```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

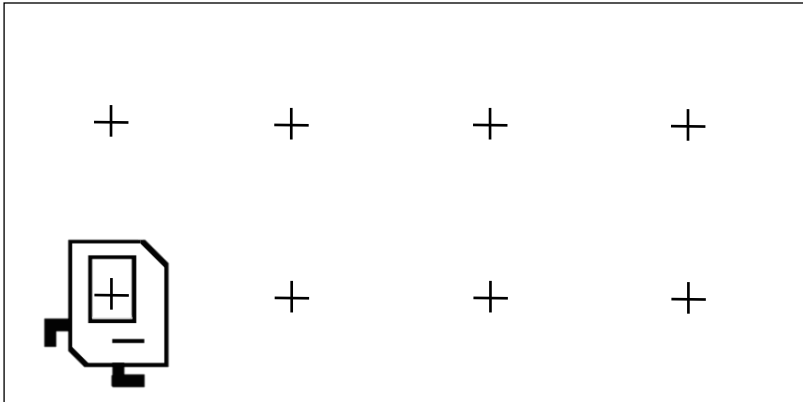
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```


Fencepost

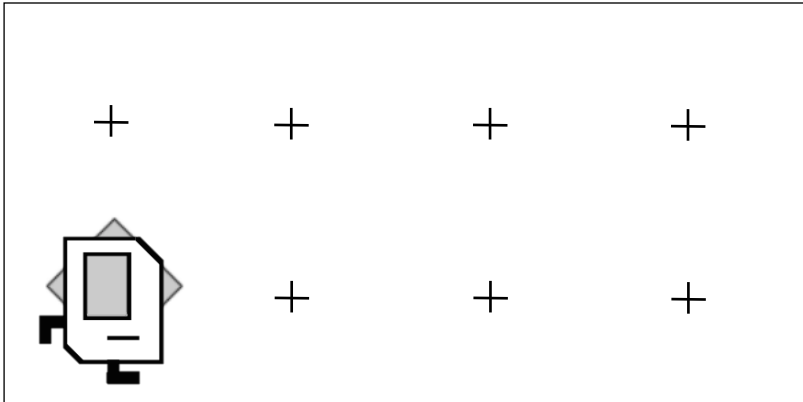
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

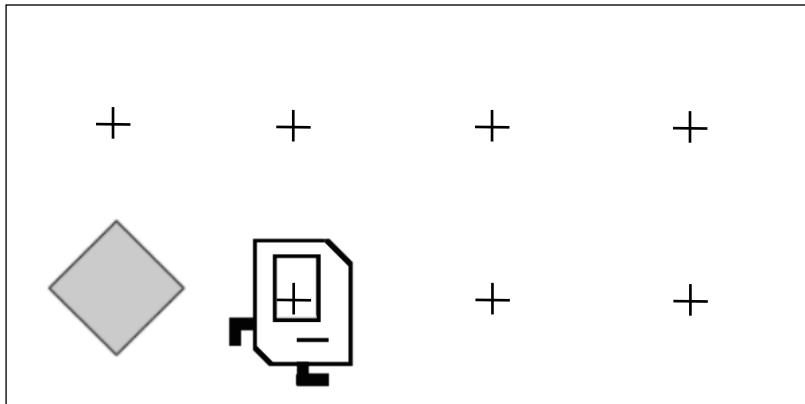
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

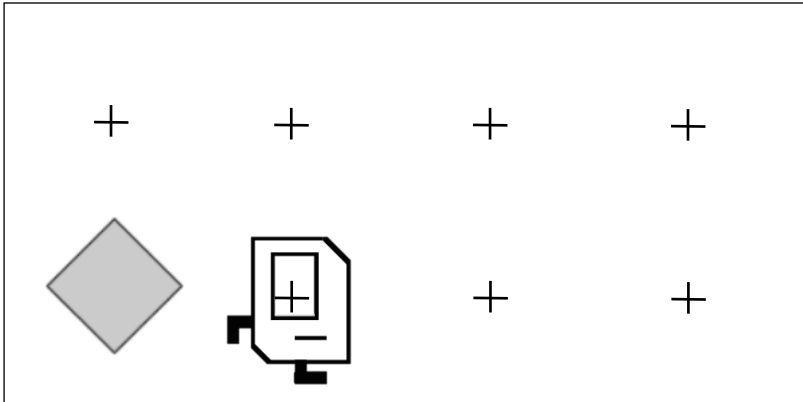
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

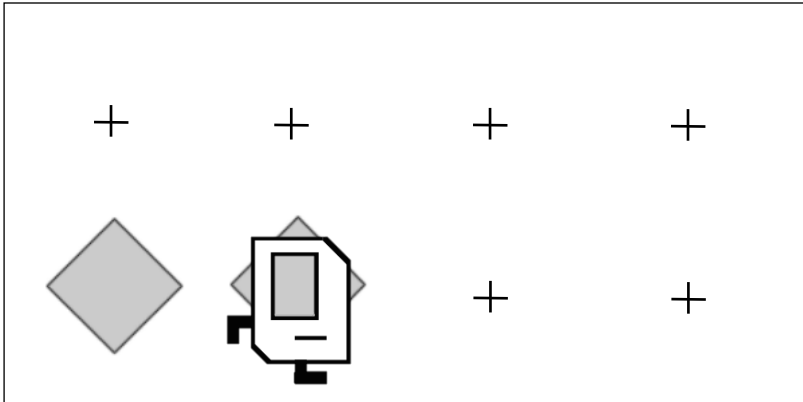
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

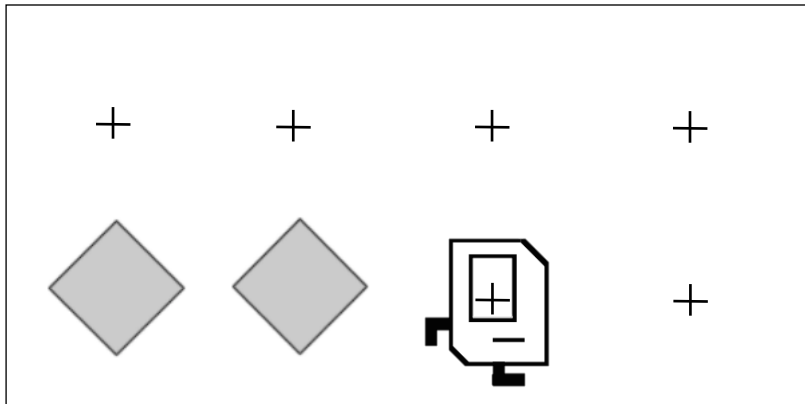
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?

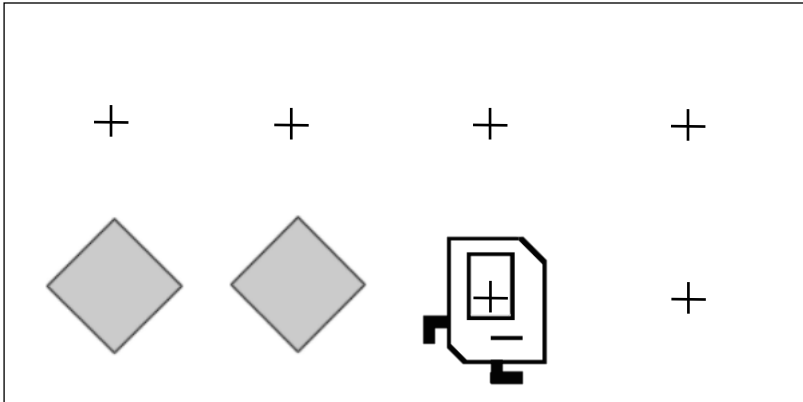


```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

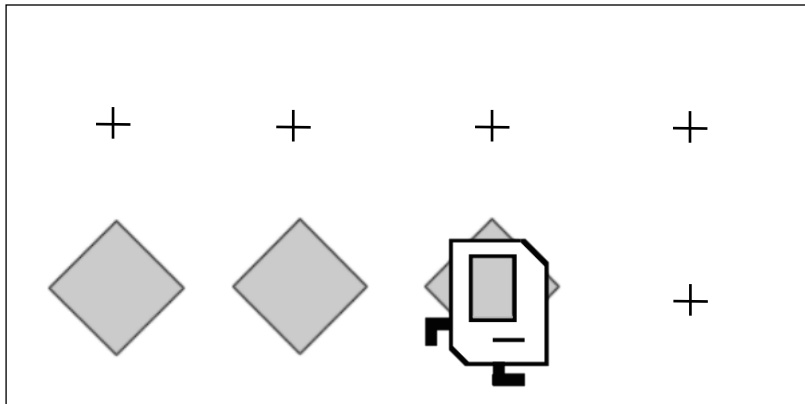
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?

```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```



Fencepost

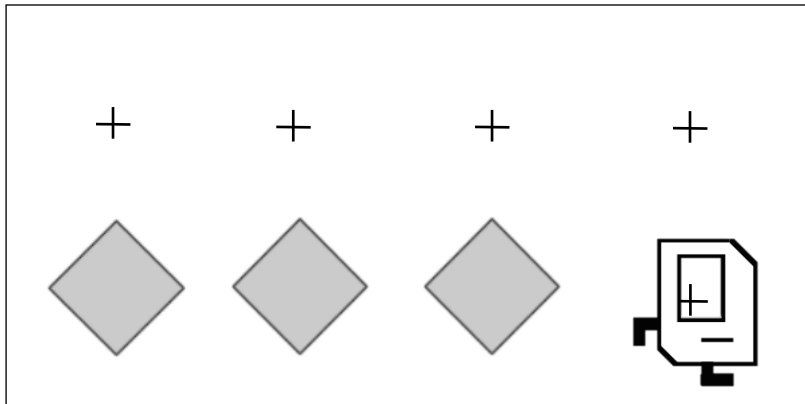
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```


Fencepost

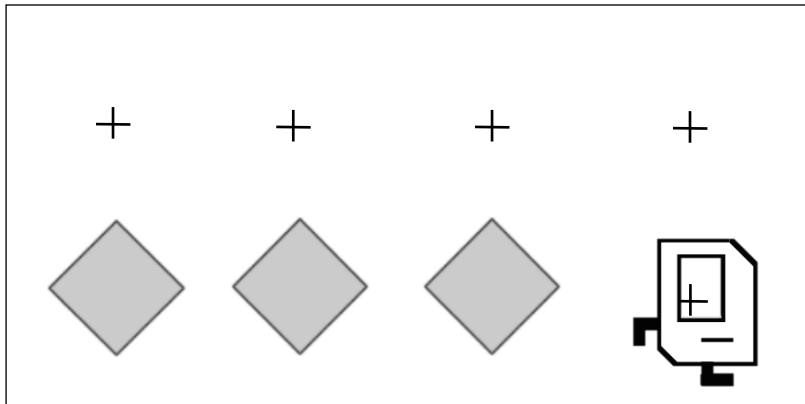
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

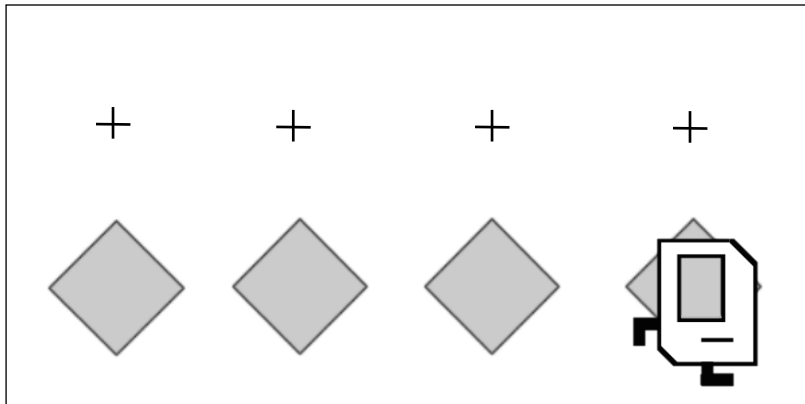
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

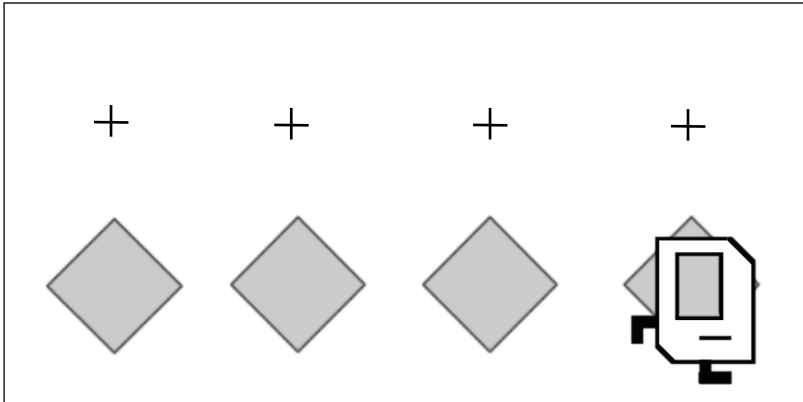
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

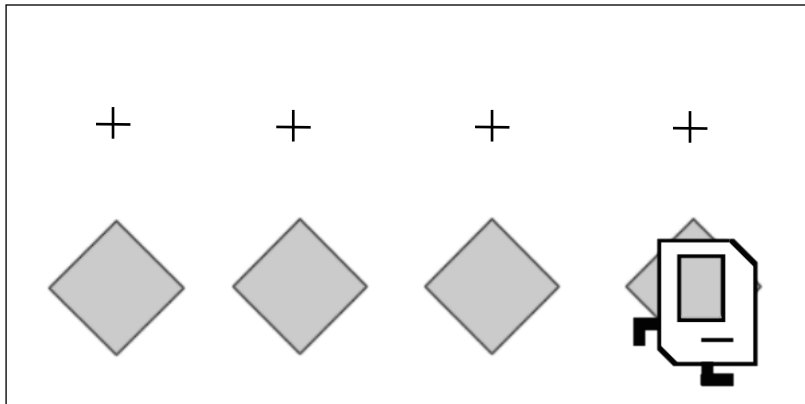
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
while front_is_clear():  
    put_beeper()  
    move()  
put_beeper()
```

Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



We must put N
beepers but
move N-1 times!

```
put_beeper()
```

```
move()
```

```
put_beeper()
```

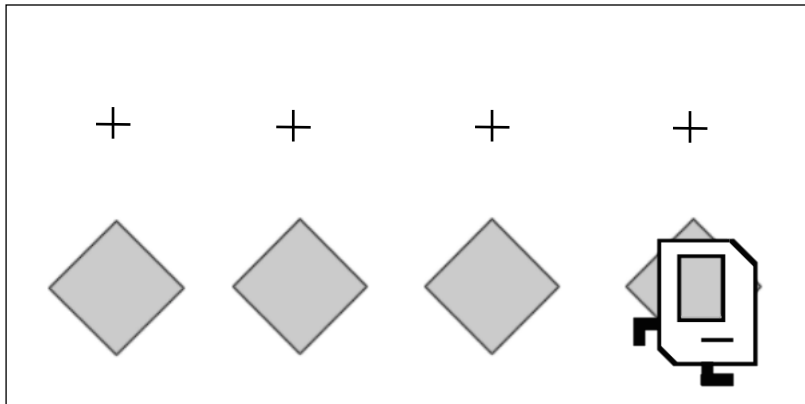
```
move()
```

```
...
```

```
put_beeper()
```

Fencepost

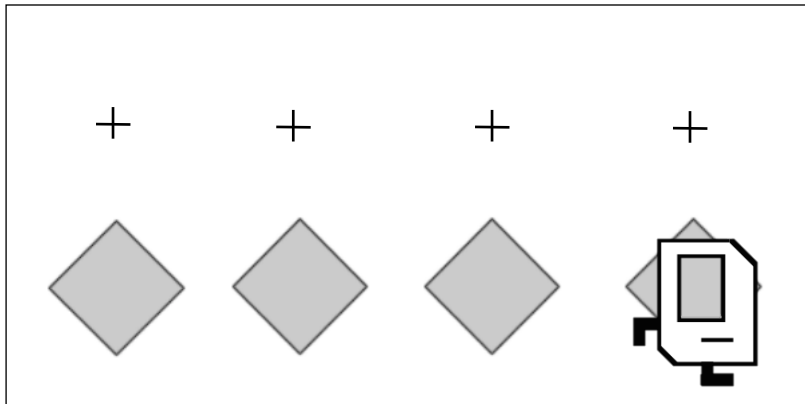
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

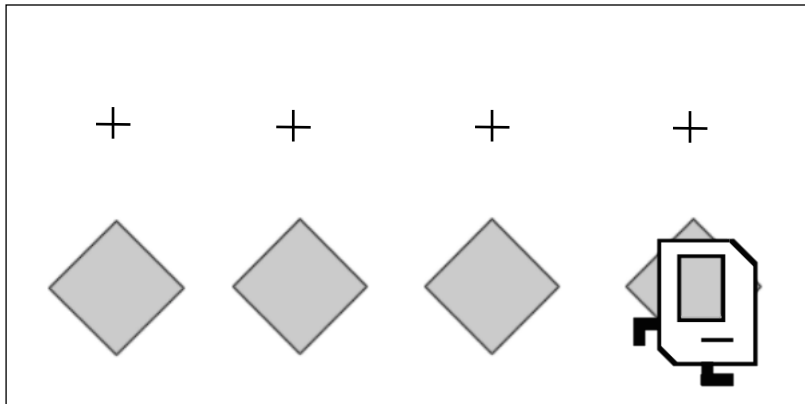
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()
```

Fencepost

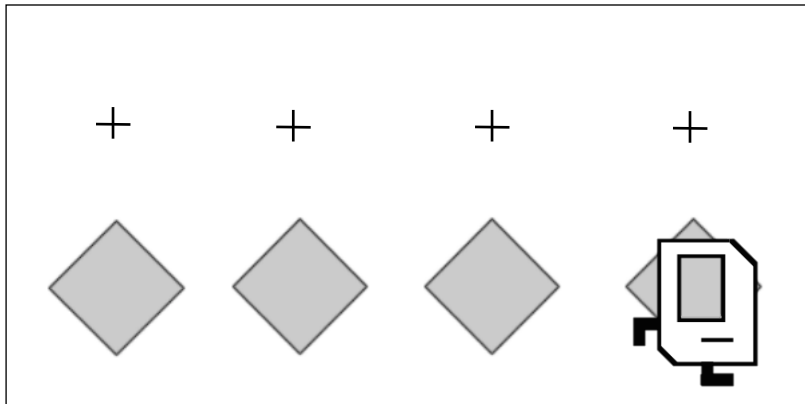
I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



```
put_beeper()  
while front_is_clear():  
    move()  
    put_beeper()
```


Fencepost

I want Karel to put down a row of beepers until it reaches a wall.
How do I do this?



We must put N
beepers but
move N-1 times!

```
put_beeper()  
move()  
put_beeper()  
move()  
put_beeper()  
...  
move()  
put_beeper()
```

Fencepost Problem



8 fence segments, but 9 posts!

Fencepost Structure

The fencepost structure is useful when you want to loop a set of statements but do one part of that set 1 *additional* time.

```
put_beeper( )           # post
while front_is_clear():
    move( )              # fence
    put_beeper( )        # post
```

```
# or...
while front_is_clear():
    put_beeper( )        # post
    move( )              # fence
put_beeper( )            # post
```

Indentation

Karel is *very* picky about indentation.

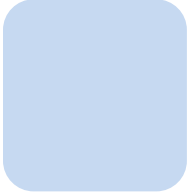
Make sure to indent a code block 1 level further when you:

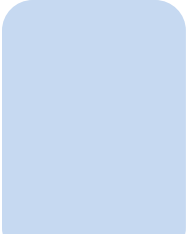
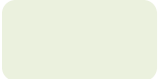
- Define a new Karel command
- Write a for loop
- Write a while loop

You may nest these. Make sure you keep track of your indentation!

Indentation

Karel is *very* picky about indentation.

```
for i in range(count):  
     statements           # note indenting
```

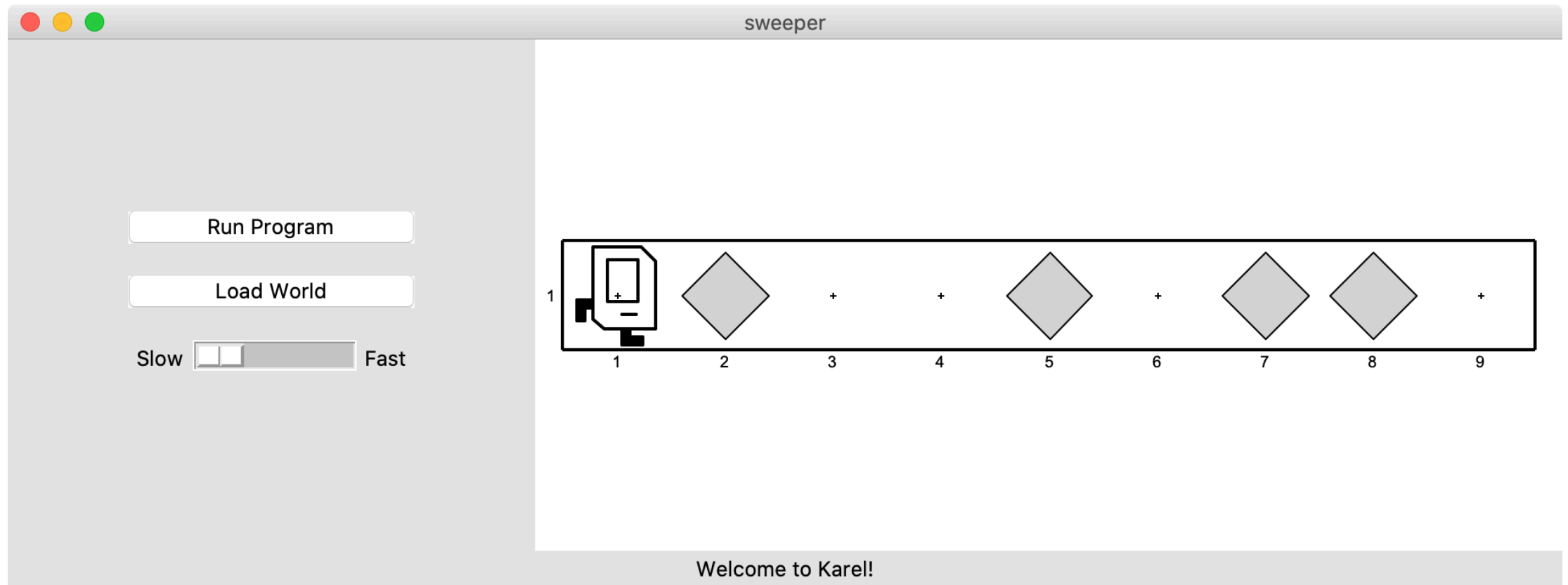
```
def my_command():  
     for i in range(3):  
         turn_left()  
        put_beeper()
```

Lecture Plan

- Review: Karel and Control Flow
- **If/Else Statements**
- Practice: Hurdle Jumper
- Decomposition and Top-Down Design

If Statements

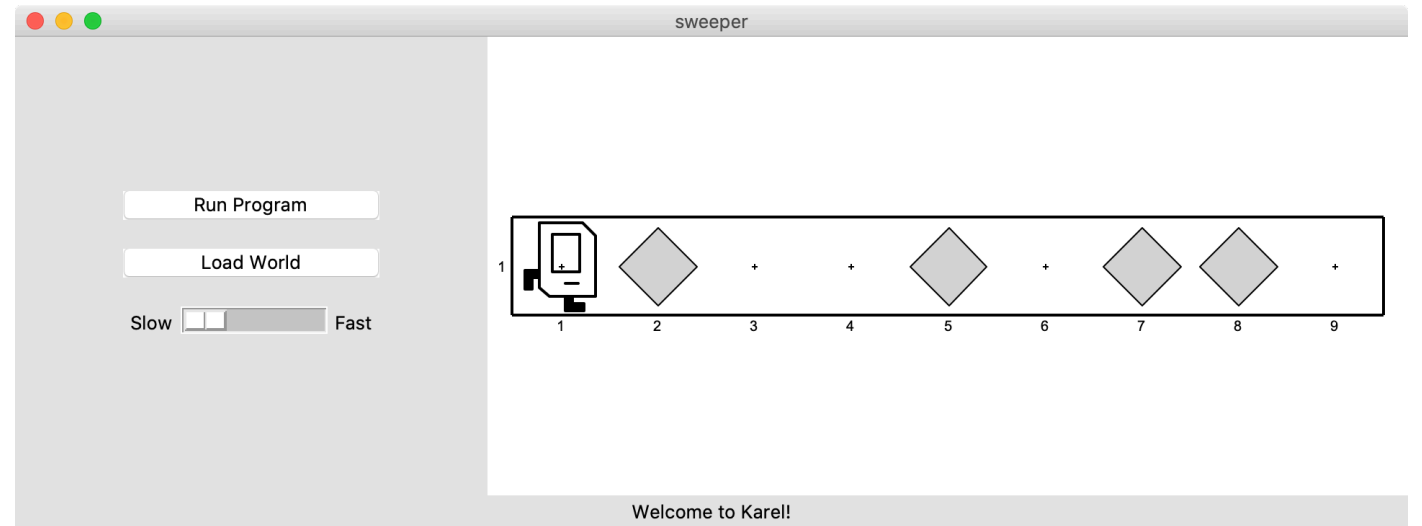
I want to make Karel clean up all beepers in front of it until it reaches a wall. How do I do this?



If Statements

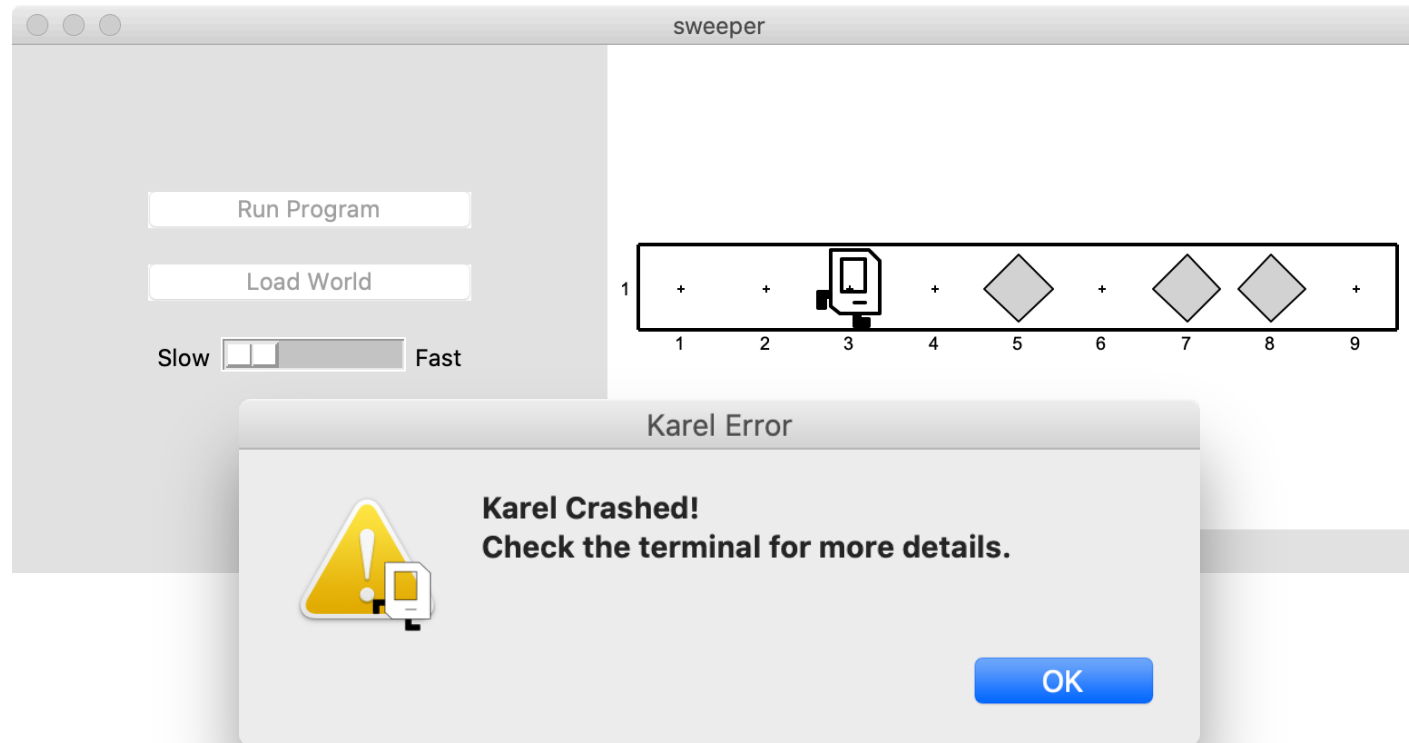
Will this work?

```
while front_is_clear():  
    move()  
    pick_beeper()
```



No. This may crash, because Karel *cannot pick up beepers if there aren't any*. We don't **always** want Karel to pick up beepers; just when there is a beeper to pick up.

If Statements



```
/usr/local/bin/python3.8 /Users/nicktroccoli/Developer/csbridge-sandbox/starter/Lecture3/sweeper.py
Traceback (most recent call last):
File "/Users/nicktroccoli/Developer/csbridge-sandbox/starter/Lecture3/sweeper.py", line 17, in main
    pick_beeper()
KarelException: Karel crashed while on avenue 3 and street 1, facing East
Invalid action: Karel attempted to pick up a beeper, but there were none on the current corner.
```

If Statements

Instead, use an **if** statement:

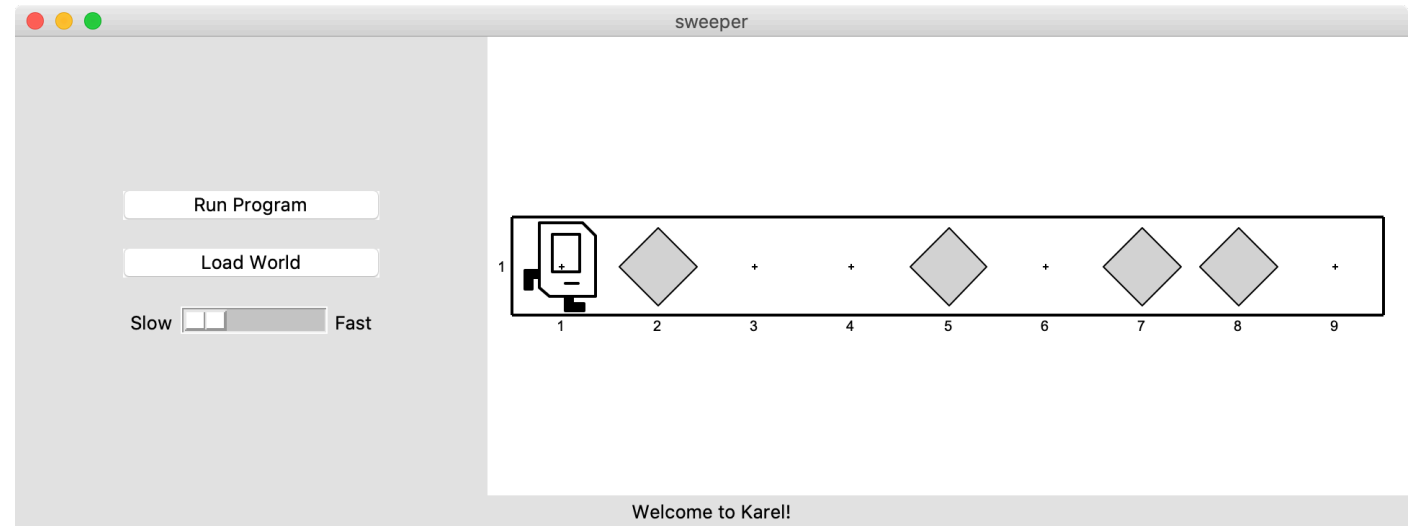
```
if condition:  
    statement  
    statement  
    ...
```

Runs the statements in the body *once* if ***condition*** is true. These are the same conditions you can use for **while** loops!

If Statements

Now we can say:

```
while front_is_clear():  
    move()  
    if beepers_present():  
        pick_beeper()
```



Karel won't crash because it will only pick up a beeper if there is one.

If Statements and Indentation

if *condition*:

statements

note indenting

def `safe_pick_up()`:

if `beepers_present()`:

`pick_beeper()` # note indenting

If/Else Statements

What if we want to do one thing if some condition is true, and another otherwise? We can add an **else** statement:

```
if condition:  
    statement  
    statement  
    ...  
else:  
    statement  
    statement  
    ...
```

This will run the first group of statements if *condition* is true; otherwise, it runs the second group of statements.

If/Else Statements

What does this code do?

```
def main():  
    if beepers_present():  
        pick_beeper()  
    else:  
        put_beeper()
```

If/Else Statements and Indentation

```
if condition:  
    statements          # note indenting  
else:  
    statements          # note indenting
```

```
def invert_beeper():  
    if beeper_present():  
        pick_beeper() # note indenting  
    else:  
        put_beeper()  # note indenting
```

Karel and Control Flow

Congratulations! You've learned all of control flow in Karel.

Control Flow lets us control the “flow” of our Karel program. For example, repeat something more than once, or only do something in certain cases.

Want to repeat something? Use a **for** or **while** loop.

- **for** if we know how many times
- **while** if we don't know how many times

Want to conditionally do something? Use **if** (with an optional **else**)

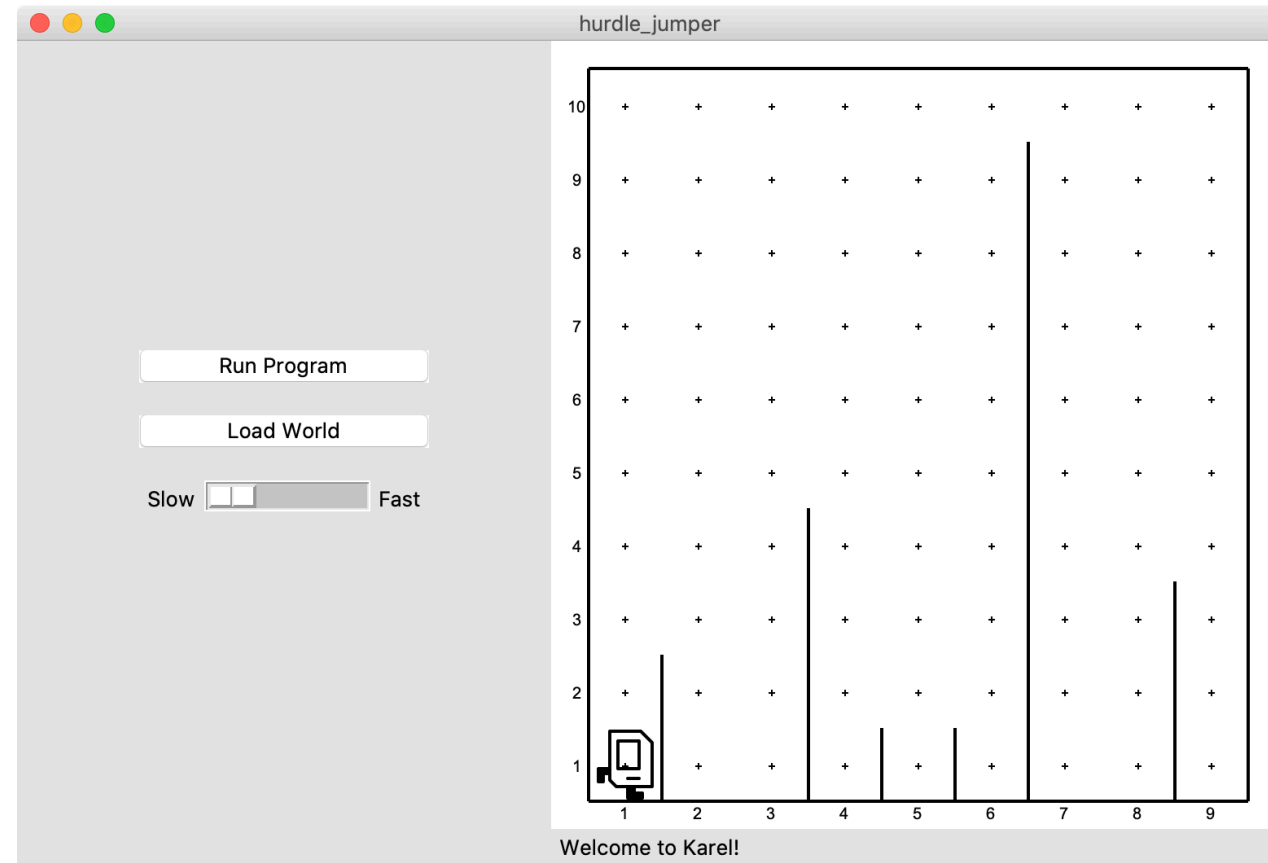
Lecture Plan

- Review: Karel and Control Flow
- If/Else Statements
- **Practice: Hurdle Jumper**
- Decomposition and Top-Down Design

Hurdle Jumper

Karel is in the Olympics! We want to write a Karel program that hops hurdles.

- Karel starts at (1,1) facing East and should end up at the end of row 1 facing east.
- The world has 9 columns.
- There are an unknown number of "hurdles" (walls) of varying heights that Karel must ascend and descend to get to the other side.



Hurdle Jumper

Demo

Hurdle Jumper

Precondition: something you *assume* is true at the start of a function or code block

Postcondition: something you *promise* is true at the end of a function or code block

Pre/post-conditions should be documented using comments.

```
def jump_hurdle():  
    """  
    Karel jumps over one hurdle of arbitrary height.  
    Pre-condition: Karel is facing east next to a hurdle.  
    Post-condition: Karel is facing east at the bottom of  
                    the other side of the hurdle.  
    """  
    ascend_hurdle()  
    move()  
    descend_hurdle()
```

Hurdle Jumper

Precondition: something you *assume* is true at the start of a function or code block

Postcondition: something you *promise* is true at the end of a function or code block

Pre/post-conditions should be documented using comments.

```
def main():  
    for i in range(8):  
        # What do we assume is true here?  
        if front_is_clear():  
            move()  
        else:  
            jump_hurdle()  
        # Does the postcondition match?
```

Lecture Plan

- **Review:** Karel and Control Flow
- If/Else Statements
- **Practice:** Hurdle Jumper
- **Decomposition and Top-Down Design**

Decomposition

- Breaking down problems into smaller, more approachable sub-problems (e.g. our own Karel commands)
- Each piece should solve **one** problem/task (< ~ 20 lines of code)
 - Descriptively-named
 - Well-commented!
- E.g. getting up in the morning:
 - Wake up
 - Brush teeth
 - Put toothpaste on toothbrush
 - Insert toothbrush into mouth
 - Move toothbrush against teeth
 - ...
 - ...

Top-Down Design

- Start from a large task and break it up into smaller pieces
- Ok to write your program in terms of commands that don't exist yet
- **Goal:** make our programs easily readable by humans
 - Commenting
 - Decomposition
- E.g. Hurdle Jumper

Lecture Recap

- **Review:** Karel and Control Flow
- If/Else Statements
- **Practice:** Hurdle Jumper
- Decomposition and Top-Down Design

Karel Resources

CS Bridge

Resources ▼

Projects ▼

Karel Reader

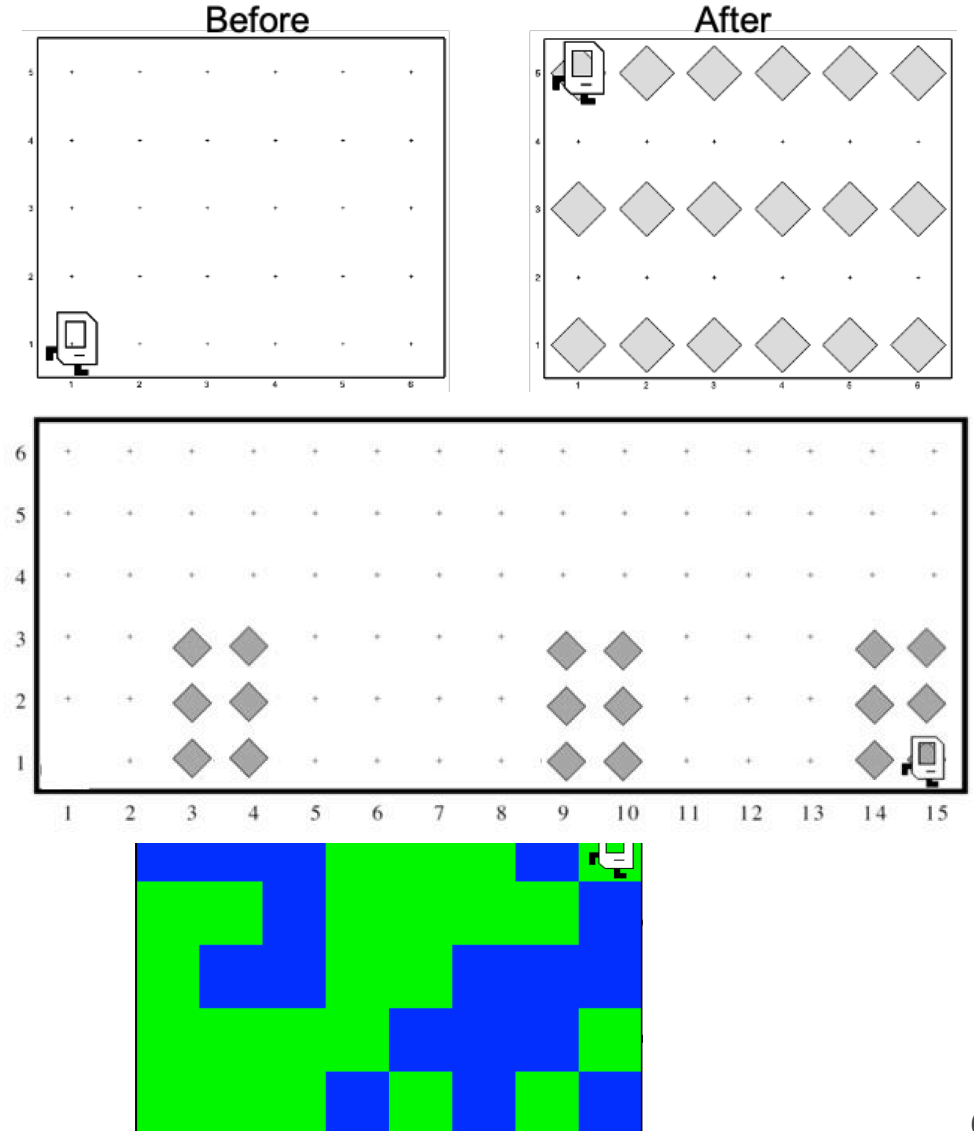
Karel Reference



1

Rest Of Today

- **Quickstart:** Implement a program where Karel draws stripes with Beepers.
- **Section:** Implement a program where Karel builds Hospitals
- **Project:** Write a program where Karel paints any world randomly with green and blue squares.



What's Next?

- Time for your section's quickstart time!
- Check your section's Ed group for more information